Feasibility Study on the Use of a Genetic Algorithm for the Generation of Air Traffic Scenarios Based on Recorded Field Data

Robert D. Oaks

January 27, 2000

Traffic Flow Management Branch, ACT-250
William J. Hughes Technical Center
Atlantic City, New Jersey 08405

Table of Contents

List of Figures

List of Tables

Distribution List

This document, published by FAA ACT-250, was distributed to the following members of the organizations listed below. Although the final distribution may not be limited to the participants below, as of the publish date of this document, the list constitutes its main distribution.

Lockheed Martin
Glenn Hahn
Steve Kazunas
Ed McKay
George Loffredo
Berk Sensoy

FAA, AUA-200
Jesse Wijntjes

AST
Duane Ball
Gary Wright

MITRE CAASD
Dan Brudnicki
William Arthur

Executive Summary

The Traffic Flow Management Branch (ACT-250) at the FAA's William J. Hughes Technical Center (WJHTC) in Atlantic City, NJ, was tasked to develop scenarios of realistic air traffic for the Accuracy Acceptance Testing of the User Request Evaluation Tool Core Capability/Limited Deployment (URET CCLD). The scenarios are required to include a specified quantity of aircraft-to-aircraft and aircraft-to-airspace encounters. In order to create these encounters ACT-250 will time shift the recorded field data. Time shifting means to change the start time of a flight, while retaining the relative times of all other flight related events.

The goal of this study was to determine if a Genetic Algorithm could be used to modify the recorded flight start times of flights so the desired number of encounters was created. In this study an implementation of the Genetic Algorithm was developed to solve a simplified version of ACT-250's actual scenario generation task.

It was found that a Genetic Algorithm was able to solve the simplified problem. It was able to time shift flights causing the desired number and distribution of aircraft-to-aircraft encounters; and was able to accomplish this with a minimal amount of air traffic data. The primary concern regarding this implementation is the time that it may take to solve the problem. Three approaches to reduce the running time of the program were identified:

- to distribute the software processes,
- to consider alternative implementation methods, and
- to optimize the input parameters.

Each of these suggestions is being further investigated by ACT-250.

1. Introduction

1.1 Background

The Federal Aviation Administration (FAA) has contracted Lockheed Martin Corporation Air Traffic Management Division (LMATM) to develop and deploy a Conflict Probe Decision Support Tool. The deployment is limited to seven En Route Air Traffic Control Centers to meet the FAA’s Free Flight Phase 1 objective. This limited deployment of the Conflict Probe application is called User Request Evaluation Tool Core Capability Limited Deployment (URET CCLD) and is based on the URET Daily Use system developed by MITRE's Center for Advanced Aviation System Development (CAASD) and installed in the Indianapolis and Memphis centers.

The FAA has tasked the Traffic Flow Management Branch (ACT-250) at the FAA's William J. Hughes Technical Center in Atlantic City, NJ, to develop air traffic scenarios based on recorded field data. These scenarios will be used to verify the accuracy requirements of URET CCLD.

AOS-610, in conjunction with ACT-250, ACT-230, and MITRE, collected recorded air traffic data from the Indianapolis and Memphis Air Route Traffic Control Centers (ARTCCs). Since these scenarios will have to include non-realistic aircraft-to-aircraft encounters, the field data will need to be modified. ACT-250 will modify this data by shifting the start times of aircraft flights and possibly by cloning selected flights. These modifications will be made to create encounters between the aircraft in the test scenarios, while maintaining the actual routes and profiles the aircraft originally flew.

1.2 Purpose

Although it would be possible to create these scenarios iteratively by changing the flight start times manually, it is desirable to calculate these changes algorithmically. While researching various random search techniques, ACT-250 determined that a Genetic Algorithm might be applicable. The following quote about the applicability of the use of Genetic Algorithms in general, was found to be quite appropriate to ACT-250’s problem:

> *. . . if the space to be searched is large, is known not to be perfectly smooth and unimodal (i.e. consists of a single smooth ‘hill’), or is not well understood, or if the fitness function is noisy, and if the task does not require a global optimum – i. e., if quickly finding a sufficiently good solution is enough – a GA will have a chance of being competitive with or surpassing other ‘weak’ methods (methods that do not use domain-specific knowledge in their search procedure).” (Mitchell, 1998)*

Consequently, ACT-250 initiated as study to investigate the following question: *Can a Genetic Algorithm be used to modify recorded field data so the resulting scenario meets certain constraints?*

1.3 Scope

This report documents ACT-250's test implementation of a Genetic Algorithm. Its objective was to determine if a Genetic Algorithm could be used to modify recorded field data so the resulting scenarios meet certain constraints.

1.4 Document Organization

The next two sections provide background information, with Section 2 describing air traffic scenarios and Section 3 describing the Genetic Algorithm. Section 4 describes the implementation of the Genetic Algorithm used in this study. Section 5 presents the results of using this implementation to solve example problems. The study's conclusions are presented in Section 6. Finally, Sections 7 and 8 contain a list of acronyms and references, respectively.

2. Field Data Based Air Traffic Scenarios

2.1 Air Traffic Scenarios

ACT-250 has developed a laboratory capability to support the integration and testing of air traffic management systems. Simulations and other activities conducted in this laboratory are used in the development, technical assessment, and field evaluation of emerging air traffic management decision support tools.

The air traffic scenarios, used for these simulations, are data files describing the flow of aircraft traffic through an airspace over a period of time. In the TFM Laboratory these airspaces are generally those defined for Terminal Radar Approach Control facilities (TRACONs), such as those that manage arrivals and departures around New York and Dallas/Fort Worth, and Air Route Traffic Control Centers (ARTCCs), that manage air traffic as it crosses the country.

These data files contain planning/advisory information and radar track data.

- **Planning/advisory information** describe an aircraft's planned flight. This includes, for example, flight plan messages, which present the flight's intended flight path, and interim altitude messages, which represent air traffic controller clearances for altitude changes.

- **Radar Track data** represents an aircraft's actual flight path. This consists of a series of time stamped, 3-dimensional points; each point having a latitude, a longitude, an altitude, and an associated time.

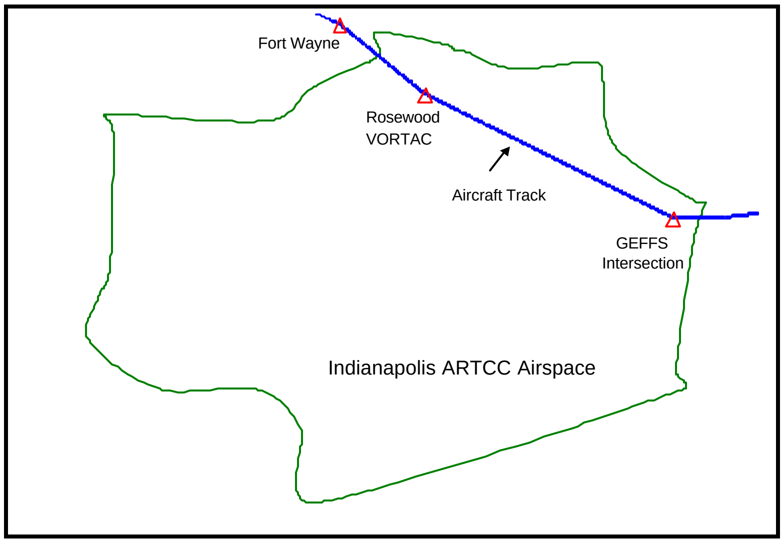

Figure 1: Example of an Aircraft Track

Figure 1 and Figure 2 present an example of recorded field data. The irregular shape in these figures represents the airspace boundary, which in this case is the Indianapolis ARTCC. The aircraft's route and its recorded position data appear as a solid line in Figure 1. The individual track points are seen more clearly in Figure 2. Neither of these figures shows the time or altitude components associated with a track. The planned route for this track begins on the right side of each of these figures. The recorded flight plan information shows that the aircraft planned to fly to the GEFFS Intersection, then to turn northwesterly toward the Rosewood VORTAC, and then to continue on to Fort Wayne.

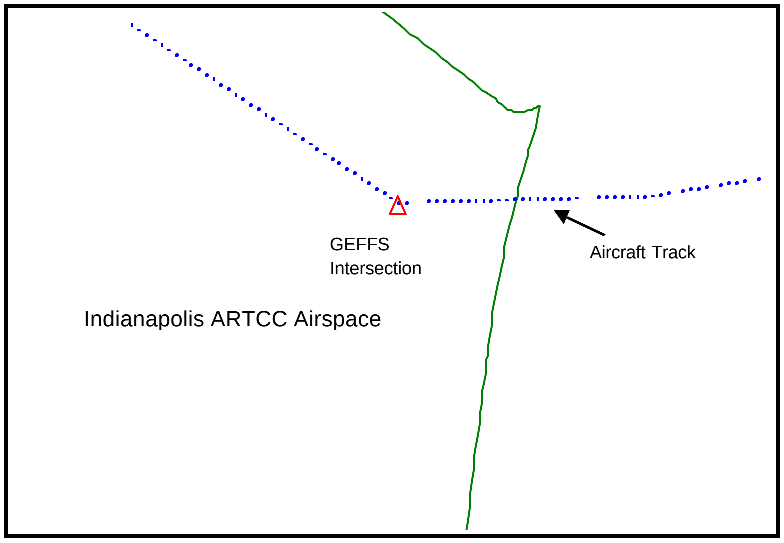

Figure 2: Blow-up of the Example Aircraft Track

2.2 Playback Scenarios

One form of these scenario files is a playback scenario. These generally utilize proprietary formats specific to a decision support tool. Usually, but not necessarily, a playback scenario is created from recorded field data. For example, the User Request Evaluation Tool (URET) is a decision support tool developed by MITRE/CAASD for use in an ARTCC. To support its development MITRE implemented the capability of playing recorded field data through the tool. Similarly the National Aeronautics and Space Administration/Ames Research Center (NASA/ARC) developed the Center TRACON Automation System (CTAS), which provides a similar capability, but uses a different format. This recorded field data for both of these tools is the intercomputer messages containing information such as flight plan data and track data.

During algorithm development and functional testing it is often adequate to use artificial scenarios. However, playback scenarios created from recorded field data are needed in evaluating decision support tools because they provide the most realistic simulation environment. Often air traffic controllers from the facility at which a tool is being considered for deployment participate in simulations to evaluate the algorithm in a field environment. These controllers prefer a simulation environment that is as close to their live environment as possible.

Frequently it is necessary to modify the scenarios. Two examples of this need are to test systems against anticipated growth and to test the capabilities of decision support tools.

- **Anticipated growth -** The FAA projects there will be a steady increase in the number of aircraft handled in ARTCCs; from 41.4 million aircraft handled in 1997 to an estimated 61.8 million in 2020. (FAA, 1996) Simulations based on increasing current traffic flows to the projected workloads are necessary to test the capabilities of air traffic control systems.

- **Capability Testing -** Decision support tools, such as those being developed by MITRE/CAASD and NASA/ARC, are usually developed using test data. In order to ensure their operational capability these tools also need to be tested with field data. ACT-250 has used field data in two recent studies. The URET Conflict Prediction Study (Cale, 1998) used simulation scenarios based on field data recorded at the Indianapolis ARTCC. The URET/CTAS Trajectory Accuracy study (Paglione, 1998) used playback scenarios based on field data recorded at the Indianapolis ARTCC for URET and Fort Worth ARTCC for CTAS.

2.3 URET/CCLD Accuracy Scenarios

The scenarios ACT-250 will develop for URET CCLD accuracy testing will be playback scenarios based on recorded field data. The URET CCLD System Specification Document (SSD) (FAA, 1998) details the requirements for the accuracy test. For example the SSD states a maximum value for the average closest point of approach (CPA) and minimum value for the standard deviation of CPA, where CPA is the distance at which two aircraft are at minimum separation from one another. Since field data is recorded at an operational ARTCC, no aircraft-to-aircraft conflicts will be present. However, in order to test these tools, conflicts are necessary. These conflicts will be created by modifying the start times of each aircraft causing the aircraft to fly the same track, but starting at a different time will do this. This time modification is done through time shifting.

Time shifting means to change the start time for a flight while retaining the relative times of all other flight related events. ACT-250 has investigated two time shifting techniques: time compression and random time adjustment.

- **Time compression** means to multiple the difference between a track's start time and some arbitrary time base by a multiplier.

- **Random time adjustment** means to randomly display a track's start time using either a uniform or normal frequency distribution.

In preliminary studies ACT-250 concluded that a combination of these two techniques can effectively create scenarios with sufficient numbers of conflict encounters. However, future constraints (such as requiring aircraft to be flying along an ATC cleared route) may require an unacceptable amount of playback scenarios in order to create a sufficient number of encounters. In addition, although these techniques create the required number of conflict encounters, the distribution of these encounters can only be achieved through trial and error. An algorithm, such as the Genetic Algorithm, that derives individual delta times that can be added to (or subtracted from) the original flight start times in order to achieve the required quantity and distribution of encounters is desired.

3. Genetic Algorithm

3.1 Introduction to Genetic Algorithms

John Holland invented Genetic algorithms (GAs) at the University of Michigan in the 1960s and 1970s and are the most prominent example of evolutionary programming. Two sources for information regarding the history of Genetic Algorithms and evolutionary programming are the following books:

- *Genetic Algorithms in Searching, Optimization, and Machine Learning*. This book presents a history of GAs along with a comprehensive information regarding the study of GAs and the application of GAs. (Goldberg, 1989)

- *An Introduction to Genetic Algorithms*. This book presents a history of Evolutionary Computation along with a good introduction to the Genetic Algorithm. (Mitchell, 1998)

There is no specific GA; instead GAs are a class of algorithms which derive their behavior from a metaphor of the processes of evolution. As such GAs are an approach to solving a problem. All GAs do have a number of traits in common. These are: a population of chromosomes, a fitness function that assigns a score to each chromosome, some form of selection according to fitness, crossover to produce new offspring, and random mutation. These traits will be presented in the following subsections through the use of a simple program example. The pseudocode for this program is presented in Figure 3.

Consider a program that is to generate a ten-character string matching the known string "HELLOWORLD". Assuming that the character string is restricted to uppercase alphabetic characters, the solution space for this problem consists of all possible ten-character strings or 26^{10} strings. A program that tests each of these possible solutions would be time prohibitive; however, a program implementing a GA can solve this problem effectively. In fact, in a computer program implementing this example the known string was generally found after about 1000 iterations, and in as few as 687 iterations. This is significantly fewer iterations than would be required to evaluate each of the more than $1.4\text{x}10^{14}$ possible solutions.

```
Initialize the population ;
Evaluate the fitness of the population ;
While (population fitness < termination criteria)
{
  Select parents from the population ;
  Randomly mutate the offspring population’s chromosomes ;
  Set the population equal to the offspring population ;
  Evaluate the fitness of the population ;
}
```

Figure 3: Genetic Algorithm Pseudocode

3.2 Population of Chromosomes

In a GA, a chromosome is defined as an array of bits or characters that represents a potential solution to a problem. Each element of the array is analogous to a gene. The values each of these genes can assume are defined as its alleles. A population of chromosomes is therefore a subset of all solutions to the problem. Since any ten-character string represents a potential solution to the example problem, a chromosome will be defined as a ten-character string for this example.

As indicated in line 1 of the pseudocode, an initial population is required for a GA. Usually this is done by selecting a random sample from the total population. In the example, ten strings, each containing ten randomly selected characters, were selected as the initial population. Such a random population is shown in the first column of Table 1 labeled Generation #0.

Table 1: Populations in the String Matching Example

	Generation #0	Selected	Crossover	Mutated
0	KJTIKDRJHI (1)	YYLSXWSBFD	THAKWWSBFD	THAKWWSBFD (4)
1	THAKWKNKXJ (1)	THAKWKNKXJ	YYLSXKNKXJ	YYLSTKNKXJ (2)
2	KLDOAWXLTT (2)	YYLSXWSBFD	YYLSXWSBFD	YYLSXWSBFD (8)
3	QUILMTVKDI (2)	KLDOAWXLTT	KLDOAWXLTT	KLDOAWXLTT (2)
4	DPZTCUCAMD (2)	YYLSXWSBFD	YYLSXWSBFD	YYLSXWSBFD (8)
5	ISQNGGFETB (1)	QUILMTVKDI	QUILMTVKDI	QUILMTVKDI (2)
6	WODVHGSLVW (1)	YYLSXWSBFD	YYLSXWSBFD	YYLSXWSBFD (8)
7	QGKMWUHAIW (1)	YYLSXWSBFD	YYLSXWSBFD	YYLSXWSBFD (8)
8	ULGNHTTRFE (2)	YYLSXWSBFD	ULGNHWSBFD	ULGNHWSBFD (4)
9	YYLSXWSBFD (8)	ULGNHTTRFE	YYLSXTTRFE	YYLSXTTRFE (4)

3.3 Fitness Function

The fitness function in a GA produces a score, which is a measure of how well a chromosome solves the problem. The fitness of the population is evaluated at lines 2 and 9 of the pseudocode. This reflects the need to test the population fitness at the top of the while loop which extends from lines 2 through 12. The fitness of a population is usually considered to be either the average or best fitness of the chromosomes in the population.

The fitness function in the example program is based on how many of the ten characters in a chromosome match the known string. The chromosome's fitness is set equal to 2^i where i is the number of matches. The value in parenthesis in column 1 of Table 1 is the chromosome's fitness. E.g., chromosome #2 has one character in common with "HELLOWORLD" – the 'W' in character position six – therefore the fitness of chromosome #2 is equal to 2^1 or 2. In the example the average fitness of the population is 2.1.

3.4 Selection According to Fitness

All GAs have some form of selection based on fitness. Generally parent chromosomes are randomly selected from the population with the probability of selection being directly proportional to a chromosome's fitness. In the example, the second column of Table 1, labeled Selected, represents ten chromosomes randomly selected from the Generation #0 population. These were selected based on each individual chromosome's contribution to the total fitness. I.e. the probability of Chromosome #2 being selected was 0.095 (2/21, where 2 is the chromosome's individual contribution to the total fitness and 21 is the population's total fitness). This random selection was done "with replacement." This is seen in this example in that Chromosome # 9, which had the highest probability of selection (0.381 = 8/21) was selected six times.

Note that this selection process does not need to be static. Sometimes, in order to lessen the possibility of an early-occurring highly fit chromosome dominating the solution, this fitness function is scaled so less fit chromosomes have a better chance of survival in the earlier generations and the more fit chromosomes have a better chance of survival in the later generations. (Goldberg, 1989)

3.5 Crossover to Produce New Offspring

The next step in a GA – line 6 of the pseudocode – is to create an offspring population based on crossover. Crossover, which occurs according to a probability called the Probability of Crossover, causes the chromosomes of the two parents to be combined according to some rule to create new chromosomes. This is usually done by swapping the parent's bits at randomly selected locus position(s).

In our example the chromosomes were paired sequentially for crossover and a Probability of Crossover equal to 0.25 was used. In the example, this crossover rule caused the first five characters of each chromosome of the pair to be swapped. The result of crossover is the chromosomes in the column labeled Crossover. Note in this example that crossover only occurred for the first pair (i.e., selected Chromosomes #0 and #1).

3.6 Random Mutation

The chromosomes of this offspring population are then considered for mutation in which the individual bits within the new chromosome may then be individually changed with some probability called the Probability of Mutation.

In the example, each character was mutated based on the probability of 0.01. Table 1 shows that in this example only one character was mutated. This is the fifth character in Chromosome # 1 (i.e., the character 'X' was mutated to the character 'T'). The resulting population of chromosomes is shown in Table 1as the column labeled Mutated. The fitness of each of these new chromosomes is shown in parenthesis. This offspring population is then moved to the generation population and the fitness of this population is calculated. The average fitness of this new population is 5.0; however, it is noted that the one chromosome with the highest fitness is represented four times. This means that this most fit chromosome will have a higher probability of being selected as a parent as the loop is continued until the desired fitness is achieved.

4. Implementation of the Genetic Algorithm

4.1 Introduction to the Implementation

ACT-250 developed the program *Cat*[1], which is an implementation of a Genetic Algorithm that solves a simplified version of the problem they have been tasked to solve. The problem has been simplified in that Cat only uses instantaneous spatial proximity to determine encounters and encounter geometry. Other factors such as time are disregarded. *Cat* uses track data from previous studies ACT-250 has done. This track data has been interpolated and otherwise “cleaned up,” which eliminated many of the problems inherent with recorded field data.

Cat compares each track’s track points with the track points from all the other tracks to find aircraft-to-aircraft encounters. These encounters are defined as situations when the closest point of approach (CPA) between two tracks is less than 20 nautical miles in the horizontal plane and less than either 1000 or 2000 feet in the vertical axis, depending on the aircraft’s altitude. In addition to the distance at CPA, the encounter angle (EA) is calculated. This angle is defined as 180° when two flights are flying directly toward each other and 0° when two aircraft are in-trail. *Cat* tallies the occurrences in the appropriate bins using the bins defined in Table 2 and Table 3. *Cat* then uses a GA to find delta times to be subtracted from the individual track start times so that the counts in these bins are within desired lower and upper bounds. These bounds are the input parameters listed in the Low Bound and High Bound columns. These parameters are defined in Section 4.2

Bin	Low Bound	High Bound
0 nm ≤ CPA < 5 nm	DesiredD05lo	DesiredD05hi
5 nm ≤ CPA < 10 nm	DesiredD10lo	DesiredD10hi
10 nm ≤ CPA < 15 nm	DesiredD15lo	DesiredD15hi
15 nm ≤ CPA < 20 nm	DesiredD20lo	DesiredD20hi

Table 2: Closest Point of Approach (CPA) Bins

Bin	Low Bound	High Bound
0 deg ≤ EA < 30 deg	DesiredA30lo	DesiredA30hi
30 deg ≤ EA < 60 deg	DesiredA60lo	DesiredA60hi
60 deg ≤ EA < 90 deg	DesiredA90lo	DesiredA90hi
90 deg ≤ EA < 120 deg	DesiredA120lo	DesiredA120hi
120 deg ≤ EA < 150 deg	DesiredA150lo	DesiredA150hi
150 deg ≤ EA < 180 deg	DesiredA180lo	DesiredA180hi

Table 3: Encounter Angle (EA) Bins

[1] The program *Cat* was named for the character Cat on the British television series *Red Dwarf.* Cat is a humanized feline; the result of 3,000,000 years of evolution on the space ship Red Dwarf after all but one of its crew were killed by a radiation leak.Cat was developed using gcc, the GNU C/C++ Version 2.7.2.3 compiler, and libg+, the GNU C/C++ Version 2.7.2 libraries. Cat was implemented on a Sun Ultra ES-4500 400 MHz workstation using the Solaris Version 2.6 operating system.

4.2 Program Input

The following inputs are provided to *Cat*:

- **DesiredFit -** The minimum value of the fitness function which will be considered acceptable as a solution to the problem.
- **DesiredD05lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have a Closest Point of Approach (CPA) in the interval [0, 5) nautical miles. (See Table 2.)
- **DesiredD05hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have a Closest Point of Approach (CPA) in the interval [0, 5) nautical miles. (See Table 2.)
- **DesiredD10lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have a Closest Point of Approach (CPA) in the interval [5, 10) nautical miles. (See Table 2.)
- **DesiredD10hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have a Closest Point of Approach (CPA) in the interval [5, 10) nautical miles. (See Table 2.)
- **DesiredD15lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have a Closest Point of Approach (CPA) in the interval [10, 15) nautical miles. (See Table 2.)
- **DesiredD15hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have a Closest Point of Approach (CPA) in the interval [10, 15) nautical miles. (See Table 2.)
- **DesiredD20lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have a Closest Point of Approach (CPA) in the interval [15, 20) nautical miles. (See Table 2.)
- **DesiredD20hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have a Closest Point of Approach (CPA) in the interval [15, 20) nautical miles. (See Table 2.)
- **DesiredA30lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [0, 30) degrees. (See Table 3.)
- **DesiredA30hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [0, 30) degrees. (See Table 3.)
- **DesiredA60lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [30, 60) degrees. (See Table 3.)
- **DesiredA60hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [30, 60) degrees. (See Table 3.)

- **DesiredA90lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [60, 90) degrees. (See Table 3.)

- **DesiredA90hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [60, 90) degrees. (See Table 3.)

- **DesiredA120lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [90, 120) degrees. (See Table 3.)

- **DesiredA120hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [90, 120) degrees. (See Table 3.)

- **DesiredA150lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [120, 150) degrees. (See Table 3.)

- **DesiredA150hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [120, 150) degrees. (See Table 3.)

- **DesiredA180lo -** The lower constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [150, 180) degrees. (See Table 3.)

- **DesiredA180hi -** The upper constraint bound for the number of aircraft-to-aircraft encounters that have an Encounter Angle (EA) in the interval [150, 180) degrees. (See Table 3.)

- **MaxGen -** The maximum number of generations that will be allowed before terminating the program.

- **PopNbr -** The size of the population of chromosomes.

- **BaseTime -** An arbitrary time prior to the start time of the first track.

- **Pc -** Probability of Crossover.

- **Pm -** Probability of Mutation.

- **Seed -** Seed for pseudorandom number generator.

4.3 Program Processes

Cat implements the GA in the following manner:

- **Definition of the Chromosome Population** – In *Cat* a chromosome consists of a sequence of delta times (genes) and is represented as $\{\Delta t_1, \Delta t_2, \ldots, \Delta t_n\}$, where there is a delta time associated with each aircraft. For example, the chromosome {0, 750, 90, …} means to start the first flight at its original time, to start the second flight 750 seconds earlier than its original start time, to start the third flight 90 seconds earlier, etc.[2] Each gene can assume 360 values (alleles) (i.e., 0, 1, 2, …, 359) representing the delta times in tens of seconds.

Cat constructs an initial population consisting of one chromosome that represents the start times of the original field data. I.e., the chromosome {0, 0, 0, …}. The remaining chromosomes have delta times selected randomly from a uniform distribution in the interval 0 to 359. Then number of chromosomes in the population is an input parameter (*NbrPop*).

- **Definition of the Fitness Function** – *Cat*'s goal is to find a set of measured values that fall within two desired bounds (Represented as low bound and high bound in Table 2 and Table 3). The fitness function implemented in *Cat* rewards both individual and multiple instances where this occurs; while at the same time penalizing instances in which the measured value is far from either a lower or an upper bound.

  The fitness function developed to achieve these two goals is:

$$Fitness = \frac{2^f}{2^N}$$

  where

$$f = \sum_{i=1}^{N} f_i$$

  with i indicating the constraint bound, N equal to the number of constraint bounds, and

$$f_i = \begin{cases} \left(\dfrac{measured}{lowbound}\right)^2 & \text{if } measured < lowbound \\ 1 & \text{if } lowbound \le measured \le highbound \\ \left(\dfrac{highbound + lowbound - measured}{lowbound}\right)^2 & \text{if } measured > highbound \end{cases}$$

  The value of f_i provides a value of one if the measured value is between the high and low bounds and rapidly decreases to zero as the measured value gets further away from either of the bounds.

  Once the individual values of f_i are computed their sum f is evaluated as a floating point number in the interval 0 to N. The fitness is then calculated as:

$$Fitness = \frac{2^f}{2^N}$$

  Figure 4 is a graphical representation of the individual contribution of each of the constraint bins (f_i). Figure 5 is a graphical representation of how $Fitness$ varies as a function of the sums of the individual contributions (f), where the number of constraint bins is 10.

[2] Note, as time shifting is implemented in *Cat*, an aircraft can only be started earlier than its original start time.

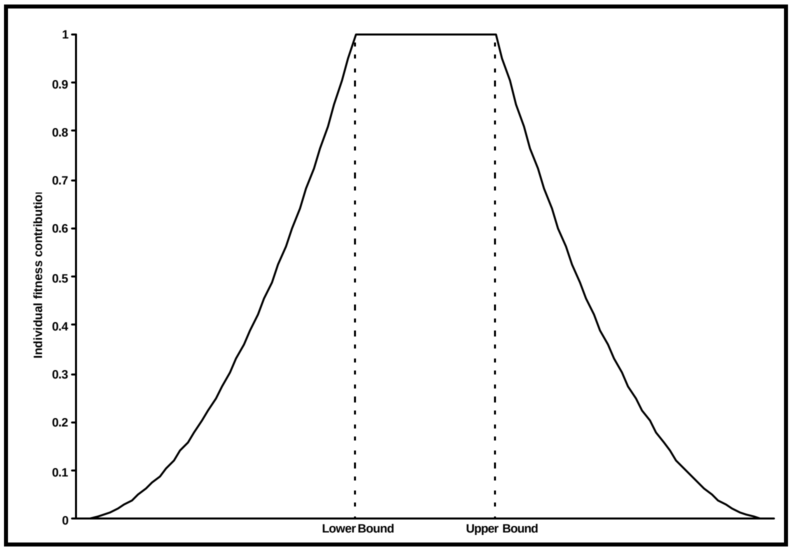

Figure 4: Individual Contribution of Constraint Bin to Fitness

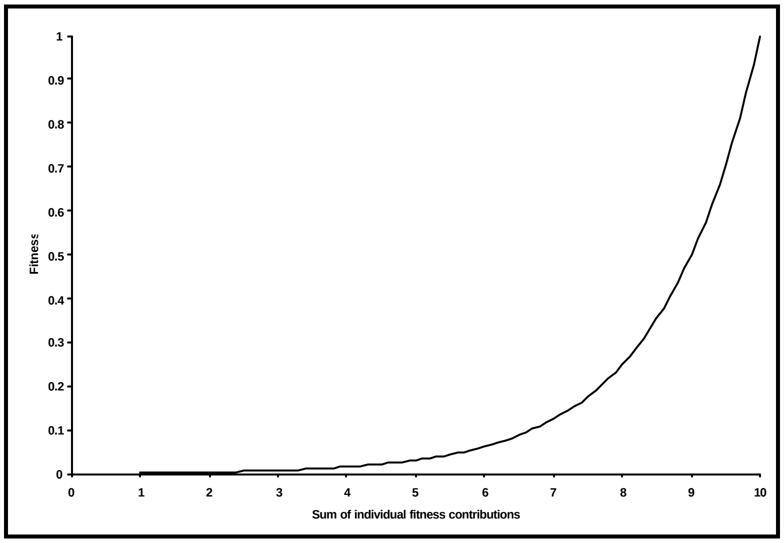

Figure 5: Graph of the Fitness Function

- **Definition of the Selection Process -** The number of parents selected is the same as the number of chromosomes in the population. The selection process uses a technique called "stochastic universal sampling," which is a different process than the selection process described in the example (Section 3.4). This technique ensures that fit chromosomes are not statistically lost in the selection process. (Mitchell, 1998, Pg. 167)

- **Definition of the Crossover Technique -** The occurrence of crossover depends on an input parameter P_c, the probability of crossover. When crossover occurs the technique uses two locus points. This technique is a different process than the crossover process described in the example (Section 3.5), which was a one-point process. In this two-point process two points are randomly selected and the segment between the two points are swapped between the two parent chromosomes.

This technique eliminates positional bias that some researchers have noted in the one-point technique. (Mitchell, 1998, Pg. 172)

- **Definition of the Mutation Technique -** The occurrence of mutation depends on an input parameter P_m, the Probability of Mutation. When mutation occurs each gene is replaced with a delta time selected randomly from a uniform distribution in the interval 0 to 359.

4.4 Program Output

The following output files are created by *Cat*:

- **Cat.log -** This ASCII log file contains information regarding the execution of the program. It is used primarily for debugging.

- **Cat.dat -** This ASCII file contains sequential information for each generation. Each line contains:
  - generation number
  - maximum fitness of a chromosome through this generation
  - average fitness in this generation
  - elapsed time since run began

- **Cat.bst** – This ASCII file contains information about the chromosome(or chromosomes) with the highest fitness.

5. Implementation Results

5.1 Introduction to Implementation Test Runs

Two series of test runs were performed. The first series were test runs to determine if *Cat* could solve the problem. The second were test runs to analyze how long it took *Cat* to repetitively solve a single problem.

All of these test runs were made using one hour of interpolated track data recorded at the Indianapolis ARTCC. The nominal results showed that this data consisted of 493 aircraft; therefore each chromosome had 493 genes. The results also showed there were nominally 344 aircraft-to-aircraft encounters.[3] The distribution of these nominal encounters for Closest Point of Approach (CPA) and for Encounter Angle (EA) is shown in Table 4.

Table 4: Nominal Encounter Counts

Measure	Nom
$0 \le CPA < 5$	20
$5 \le CPA < 10$	68
$10 \le CPA < 15$	119
$15 \le CPA < 20$	137
$0 \le EA < 30$	83
$30 \le EA < 60$	44
$60 \le EA < 90$	51
$90 \le EA < 120$	51
$120 \le EA < 150$	58
$150 \le EA < 180$	57

5.2 Ability to Solve Test

The solvability tests consisted of three runs. The first run (called Run #a) attempted to evenly distribute the encounters across the constraint bins. Distributing the 344 encounters over the five CPA bins would result in 86 encounters per bin. Distributing the 344 encounters over the six EA bins would result in about 57 encounters per bin. The input for the test run asked the program to find a solution which would result in 85 to 87 encounters in each of the CPA bins and 56 to 58 encounters in each of the EA bins. It also specified a high value of 0.99 for the desired fitness of the solution. This input is shown in Figure 6

[3] It important to note there is no correlation between these "program encounters" and "real encounters." The recorded field data used in this study had no aircraft-to-aircraft violations during the sample time.

```
#This is the Cat input file
DesiredFit 0.99
DesiredD05lo  85
DesiredD10lo  85
DesiredD15lo  85
DesiredD20lo  85
DesiredD05hi  87
DesiredD10hi  87
DesiredD15hi  87
DesiredD20hi  87
DesiredA30lo  56
DesiredA60lo  56
DesiredA90lo  56
DesiredA120lo 56
DesiredA150lo 56
DesiredA180lo 56
DesiredA30hi  58
DesiredA60hi  58
DesiredA90hi  58
DesiredA120hi 58
DesiredA150hi 58
DesiredA180hi 58
MaxGen     4000
PopNbr     20
BaseTime   46800
Pc         0.75
Pm         0.01
```

Figure 6: Listing of Cat_in.dat File Used in Run #a

Table 5: Constraint Bins in Run #a

Measure	Lo	Hi	#a
$0 \le$ CPA < 5	85	87	85
$5 \le$ CPA < 10	85	87	85
$10 \le$ CPA < 15	85	87	87
$15 \le$ CPA < 20	85	87	87
$0 \le$ EA < 30	56	58	58
$30 \le$ EA < 60	56	58	58
$60 \le$ EA < 90	56	58	56
$90 \le$ EA < 120	56	58	57
$120 \le$ EA < 150	56	58	58
$150 \le$ EA < 180	56	58	57

The results of Run #a are shown in Table 5, in which it is seen that *Cat* found a set of delta times for which the resultant encounter counts fell within all of the bins – i.e., this solution has a fitness value of 1.0. *Cat* found this solution in 173 generations and took 2955.73 seconds to solve. It must be emphasized, although there were 344 encounters in this solution, there are no assurances that these are the same 344 encounters that occurred in the nominal run. It is likely that the time shifting determined by the GA would result in different encounters involving different flight combinations.

Figure 7 presents a graph showing how the fitness of the population increases with the evolving generations of the chromosome population. The top line represents the best fitness encounter up to that generation. The bottom line represents the average fitness of the generation.

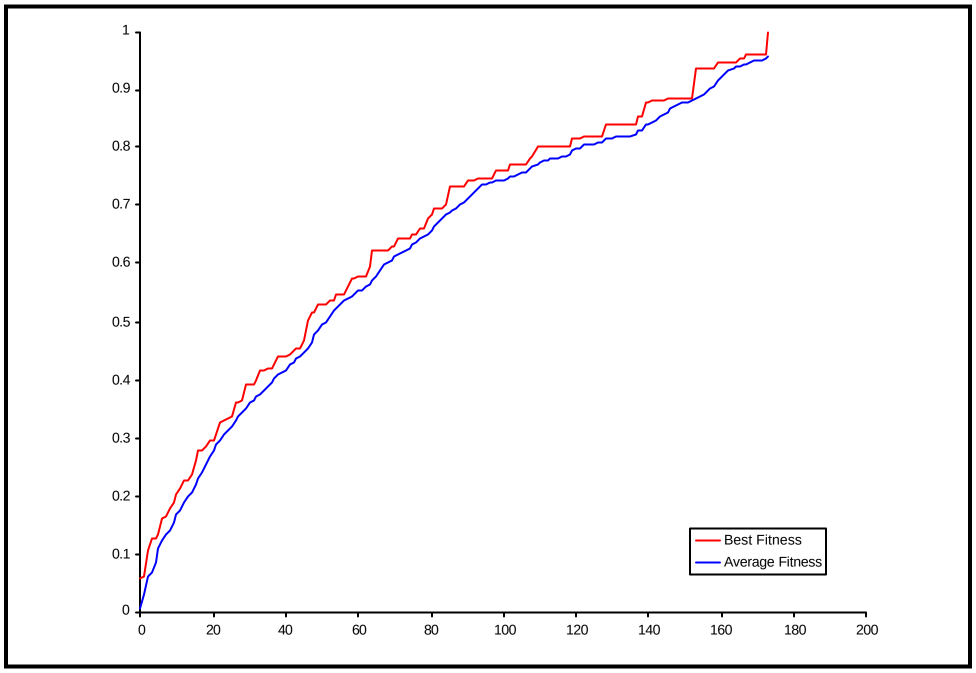

Figure 7: Fitness vs Generation

The second test to solve (Run #b) asked Cat to increase the number of encounters by 10% to 20%. These bounds are seen in the input presented in Figure 1. Note, in addition, the desired fitness was lowered to 0.975.

The results of Run #b are shown in Table 6, in which it is seen that *Cat* found a set of delta times for which the resultant encounter counts fell within all but one of the bins – it only has 63 encounters in the $120 \leq$ EA < 150 bin, which is below the low bound of 64 encounters. Run #b achieved these results in 172 generations with a best fitness of 0.978738.

```
#This is the Cat input file
DesiredFit 0.975
DesiredD05lo  22
DesiredD10lo  75
DesiredD15lo  131
DesiredD20lo  151
DesiredD05hi  24
DesiredD10hi  82
DesiredD15hi  143
DesiredD20hi  164
DesiredA30lo  91
DesiredA60lo  48
DesiredA90lo  56
DesiredA120lo 56
DesiredA150lo 64
DesiredA180lo 63
DesiredA30hi  100
DesiredA60hi  53
DesiredA90hi  61
DesiredA120hi 61
DesiredA150hi 70
DesiredA180hi 68
MaxGen     4000
PopNbr     20
BaseTime   46800
Pc         0.75
Pm         0.01
```

Figure 8: Listing of Cat_in.dat file Used in Run #b

Table 6: Constraint Bins in Run #b

Measure	Lo	Hi	#b
$0 \leq$ CPA < 5	22	24	24
$5 \leq$ CPA < 10	75	82	77
$10 \leq$ CPA < 15	131	143	133
$15 \leq$ CPA < 20	151	164	154
$0 \leq$ EA < 30	91	100	91
$30 \leq$ EA < 60	48	53	51
$60 \leq$ EA < 90	56	61	60
$90 \leq$ EA < 120	56	61	57
$120 \leq$ EA < 150	64	70	63
$150 \leq$ EA < 180	63	61	66

The third test is closer to the real problem that ACT-350 will need to solve. In this test Cat was asked to increase the number of encounters and to distribute these encounters across the bins. The input file for this test is shown in Figure 9.

The constraint bins for the solution obtained by *Cat* are shown in Table 7. The fitness for this function was 0.985589. This solution was achieved in 242 generations which took 4235.17 seconds of CPU time.

```
#This is the Cat input file
DesiredFit 0.975
DesiredD05lo  95
DesiredD10lo  95
DesiredD15lo  95
DesiredD20lo  95
DesiredD05hi  105
DesiredD10hi  105
DesiredD15hi  105
DesiredD20hi  105
DesiredA30lo  60
DesiredA60lo  60
DesiredA90lo  60
DesiredA120lo 60
DesiredA150lo 60
DesiredA180lo 60
DesiredA30hi  65
DesiredA60hi  65
DesiredA90hi  65
DesiredA120hi 65
DesiredA150hi 65
DesiredA180hi 65
MaxGen     4000
PopNbr     20
BaseTime   46800
Pc         0.75
Pm         0.01
```

Figure 9: Listing of Cat_in.dat File Used in Run #c

Table 7: Constraint Bins in Run #c

Measure	Lo	Hi	#b
0 ≤ CPA < 5	95	105	94
5 ≤ CPA < 10	95	105	95
10 ≤ CPA < 15	95	105	96
15 ≤ CPA < 20	95	105	104
0 ≤ EA < 30	60	65	65
30 ≤ EA < 60	60	65	65
60 ≤ EA < 90	60	65	64
90 ≤ EA < 120	60	65	65
120 ≤ EA < 150	60	65	65
150 ≤ EA < 180	60	65	65

5.3 Time to Solve Test

Ten runs were made to analyze the time to solve. Except for the random seed (*Seed*) for the random number generator, these ten runs had the same input, which was the same input used for Run #c. This input is shown in Figure 9.

Since the GA is a stochastic process each of the ten runs produced different results. The counts for the closest point of approach (CPA) and encounter angles (EA) for these encounters are summarized for the nominal field data and for each of the ten runs is shown in Table 8. Note that: the second test run (Run #1) had a final generation of the population containing two chromosomes with equal acceptable fitness values. The number of encounters in the constraint bins for each of these solutions is presented in the table.

Table 8: Constraint Bins in Time to Solve Test Runs

Measure	Nom	#0	#1a	#1b	#2	#3	#4	#5	#6	#7	#8	#9
$0 \le CPA < 5$	20	95	95	94	95	95	95	95	94	94	95	95
$5 \le CPA < 10$	68	96	94	96	100	95	96	95	96	96	96	94
$10 \le CPA < 15$	119	96	99	95	98	99	95	99	96	98	95	97
$15 \le CPA < 20$	137	98	97	96	94	95	102	97	95	96	101	97
$0 \le EA < 30$	83	65	64	65	65	65	65	66	65	65	66	65
$30 \le EA < 60$	44	66	65	64	65	65	62	65	63	64	64	65
$60 \le EA < 90$	51	61	65	64	65	60	65	65	61	64	64	62
$90 \le EA < 120$	51	64	62	61	66	66	66	63	64	62	65	65
$120 \le EA < 150$	58	64	64	64	64	63	65	62	64	64	64	62
$150 \le EA < 180$	57	65	65	63	65	65	65	65	64	65	64	64

Since each of the solutions consists of 493 delta times, and we were only looking for a possible solution we did not attempt to compare the solutions. We did analyze how long these runs took to achieve and acceptable solution. Figure 10 is a plot of the best fitness value as a function of elapsed CPU processing time for each of the ten runs. Each line in this graph represents the results of a different run.

Table 9 shows the time that was required to achieve a solution in each of the ten runs. A solution was achieved in as little as 1890.690 seconds and as high as 4017.100 seconds. The average time to solve the problem for these ten runs was 2993.739 seconds with a standard deviation of 628.173 seconds.

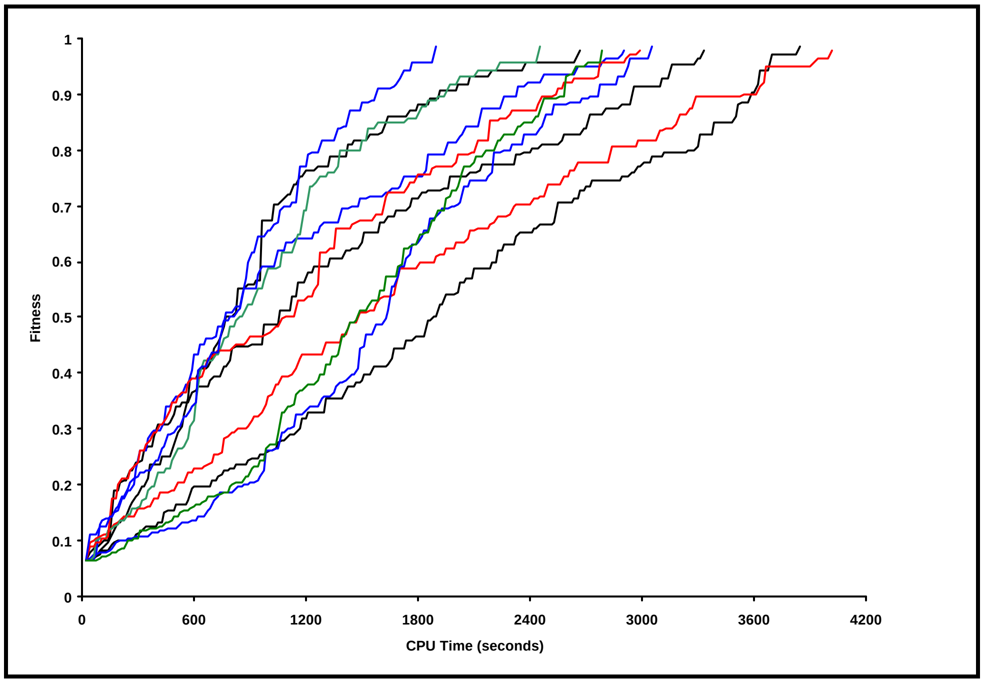

Figure 10: Best Fitness vs CPU Time

Table 9: Time to Solve in Test Runs

Run Number	Time to Solve
0	3331.14
1	3844.34
2	2787.15
3	2673.55
4	2907.13
5	4017.10
6	3051.04
7	2449.61
8	2986.64
9	1890.69

6. Conclusions

The most significant finding in this study is that a Genetic Algorithm can be used to modify recorded field data so the resulting scenario met both encounter constraints. Specifically, the start times of recorded tracks were modified so that the desired number of aircraft-to-aircraft encounters that had closest points of approach and encounter angles in specific bins was achieved. Another significant finding was that the fitness function that was designed for this implementation appears to work well. The time required to solve the simplified problem was acceptable; however, because of the additional constraints and longer playback scenarios, it is a concern if ACT-250 implements a GA to solve their task. ACT-250 should consider methods that might reduce the time to solve. These could include:

- **Distributing the software processes**. *Cat* did not take advantage of the four CPUs that are available on the processor that will be used by ACT-250. Distributing the functions between CPUs would decrease the time to solve.

- **Considering alternative methods for selection, crossover, and mutation.** Researchers have investigated many other methods. These, or extensions of them, should be considered.

- **Optimizing the input parameters**. Although preliminary tests showed there may not be a significant correlation between *PopNbr*, *Pc*, and *Pm*, it may we worth while to conduct further tests to see if these parameters affect the time to solve the problem.

In summary, the GA offers ACT-250 a systematic method to generate playback scenarios that contain the required number and distribution of constrained encounters. In addition, the GA will decrease the quantity of playback scenarios required by increasing the number of encounters within each scenario more effectively.

7. Acronyms

ACT-250	Traffic Flow Management Branch
ARC	Ames Research Center
ARTCC	Air Route Traffic Control Center
ASCII	American Standard Code for Information Interchange
CAASD	Center for Advanced Aviation Systems Development
CCLD	Core Capability/Limited Deployment
CPA	Closest Point of Approach
CPU	Central Processing Unit
CTAS	Center TRACON Automation System
EA	Encounter Angle
FAA	Federal Aviation Administration
GA	Genetic Algorithm
GNU	GNU's Not Unix[4]
Mhz	Mega hertz
NASA	National Aeronautics and Space Administration
SSD	System Specification Document
TFM	Traffic Flow Management
TRACON	Terminal Radar Approach Control Facility
URET	User Request Evaluation Tool
VORTAC	VHF Omindirectional Range Collocated with Tactical Air Navigation
WJHTC	William J. Hughes Technical Center

[4] See www.gnu.org.

8. References

Mary Lee Cale, Michael Paglione, Dr. Hollis Ryan, Dominic Timoteo, Robert Oaks, "User Request Evaluation Tool (URET) Conflict Prediction Accuracy Report," DOT/FAA/CT-TN98/8, April, 1998, FAA Technical Center, NJ.

David E. Goldberg, *Genetic Algorithms in Searching, Optimization, and Machine Learning*, Addison-Wesley, 1989.

Federal Aviation Administration, "FAA Long Range Aviation Forecasts Fiscal Years 2010, 2015, and 2020," FAA-APO-08-9, June, 1998. (The FAA issues its 12-year forecast in the spring of each year. This is used for both manpower and facility planning as well as policy analysis.)

Federal Aviation Administration, "User Request Evaluation Tool Core Capability Limited Deployment System Specification Document", FAA-ER-2929, October, 1998.

Melanie Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, MA, 1998.

Mike M. Paglione, Dr. Hollis F. Ryan, Robert D. Oaks, J. Scott Summerill, and Mary Lee Cale, “Trajectory Prediction Accuracy Report: User Request Evaluation Tool (URET)/Center TRACON Automation System (CTAS), DOT/FAA/CT-TN99/10, May, 1999.